DØ Grid Production Workshop: Introduction

Adam Lyon
Revision 1.2

1 Purpose for Workshop

The goal of this workshop is to lay the groundwork for a workable plan for DØ Grid Production for the future. The SAMGrid team has recently announced winding down development for SAMGrid, and so DØ has produced a challenging task list of items to be completed before development ceases. Manpower is limited, and so the Computing Division is looking for ways to assist the execution of the task list. To this end, the purpose of the workshop is two fold.

1. Understand the system and its components at their current state
2. Obtain input for the future of the components in the context of the DØ task list.

This is a forward looking workshop; that is the purpose is not to debate past design decisions, but rather to look to improving the future of production on the Grid. But it is important to understand some of the expectations when designing this system, and we'll do that later in section 4.

It is also important to remember that the SAMGrid system has been very successful for DØ. DØ has used this system for p17 and p20 reprocessing (billions of events each) as well as an extremely successful emergency refixing effort (1.5 billion events processed in 5 weeks; our first use of standard Grids for production). SAMGrid has been in use for MC production for several years and produces millions of events per week. SAMGrid fulfills the needs of DØ, but DØ needs increased capabilities.

2 The DØ Task List

2.1 Background

DØ relies on the "SAMGrid system" (note that "system" here means a collection of tools including SAMGrid, d0reprotools, and D0RunJob) for all remote production processing, including MC generation and reprocessing of data. Reprocessing involves running old data through a newer version of the reconstruction program to take advantage of improved algorithms and calibration constants. There is also a desire to use SAMGrid for "primary processing" (that is do the first pass processing). The goal has been to automate as much of these tasks as possible for opportunistic computing on the Grid. Indeed many of the tasks involved with MC generation and Reprocessing are automated with SAMGrid, but there are also many tasks which are still performed by hand. The

SAMGrid system is also currently not well suited to easily add functionality on short time scales.

2.2 MC Production

Currently, the SAMGrid system is used for all standard MC production. Standard production here means starting with running a Monte Carlo generator through creating final merged thumbnail data files. There are additional functions that are desired and make up this part of the DØ tasklist.

1. Currently, only the final merged thubmnails are stored to tape. There is a desire to have intermediate files also written to tape as per user supplied parameters.
2. Currently, MC production with the system must start with running the MC generator. There is a desire to start at a later MC production step, using input files already in SAM.
3. Currently, the trigger simulator is not part of the workflow, and so must be run by hand on the analysis farm. There is a desire to add "trigsim" to the workflow.
4. There are also requests to make the system more adaptable to different production paths.

2.3 (Collider Data) Production

The system right now can reprocess raw data and produce merged thumbnails. The additional functions desired are as follows:

1. Perform skimming in production. Skimming involves reading in merged thumbnails and writing out many streams of thumbnails according to physics contents. Right now this task is performed on the analysis farm.
2. Production of CAF trees. "CAF" (common analysis format) is the standard DØ root-tuple format. Currently, this task is performed by hand on the analysis farm.
3. Run `raw2sim` on minbias files. This task creates files necessary for overlaying underlying event information on top of MC events. Right now this task is performed by hand on the analysis farm.
4. Merging recocert files. Recocert files are files of root-tuples for certification. Right now, many small files are produced. There is a desire to merge them.
5. Merge more than one stream from Reco (dilepton, zerobias). The reconstruction program produces these streams along with the main data. Right now the streams are not merged, leading to small files.
6. Convert some data to AADST format. The B physics group uses a nonstandard data format (AADST). There is a desire to do this conversion on the Grid.
7. Conversion of D0reproTools to Sam V7. Essentially, make the D0ReproTools (automated submission program) compatible with the latest version of the SAM code. Some SAM code may have to change as a result.

8. Standalone production of recocert output. Recocert is run with standard production, but in case of failures, it may need to be run standalone. This is not done currently in an automated way.
9. Resource brokering. SAMGrid can submit jobs to OSG, LCG and native sites. There is no overall broker that manages submissions to these resources.
10. Fixing on the Grid. A "fixing" step involves processing old thumbnails into new thumbnail files with a program with some improved algorithm or calibration. Fixing is like reprocessing, but much faster since one does not have to go back to the Raw data. DØ sucessfully performed fixing on the Grid in early 2006. As of this time, there are no future plans for fixing, but it may be likely in the near future.
11. Analysis on the Grid. DØ does not do analysis jobs on the Grid.

3 Current State of the System

The system may be broken up into many components. This section is a very high level overview. There are three major divisions: a) components that submit SAMGrid jobs into the system, b) components and infrastructure that execute those jobs on the Grid, and c) components to run the particular application.

3.1 Submission of SAMGrid Jobs

Figure 1 shows a high level diagram for submitting jobs into the system. A SAMGrid job is a job to perform a particular task, such as process a SAM dataset with Reco (Reprocessing), create Monte Carlo with a specific card file, or merge a particular set of files. Reprocessing uses a set of scripts called `d0reprotools` to automate the creation of SAMGrid jobs (MC has their own set of scripts for doing the same thing for MC jobs, but that is not the topic of this workshop). Jobs are submitted using a SAMGrid client program, which communicates with a SAMGrid submission node.

The Submission Node is responsible for routing the job to the right place (a particular native SAMGrid cluster or the OSG/LCG grids) as well as collecting monitoring information about the jobs.

3.2 Executing jobs on the Grid

Figure 2 shows the components for executing jobs on the Grid. The submission node hands the SAMGrid job to an OSG or LCG forwarding node or a head node of a native SAMGrid cluster. This head/forwarding node determines the number of execution jobs (depends on the application) and submits them to the batch system. A feature is that the OSG and LCG grids are treated as batch systems, making this component quite generic with respect to the location of the execution site. The "batch system" transfers the execution jobs to worker nodes. The head/forwarding node can communicate job status back to the submission node.

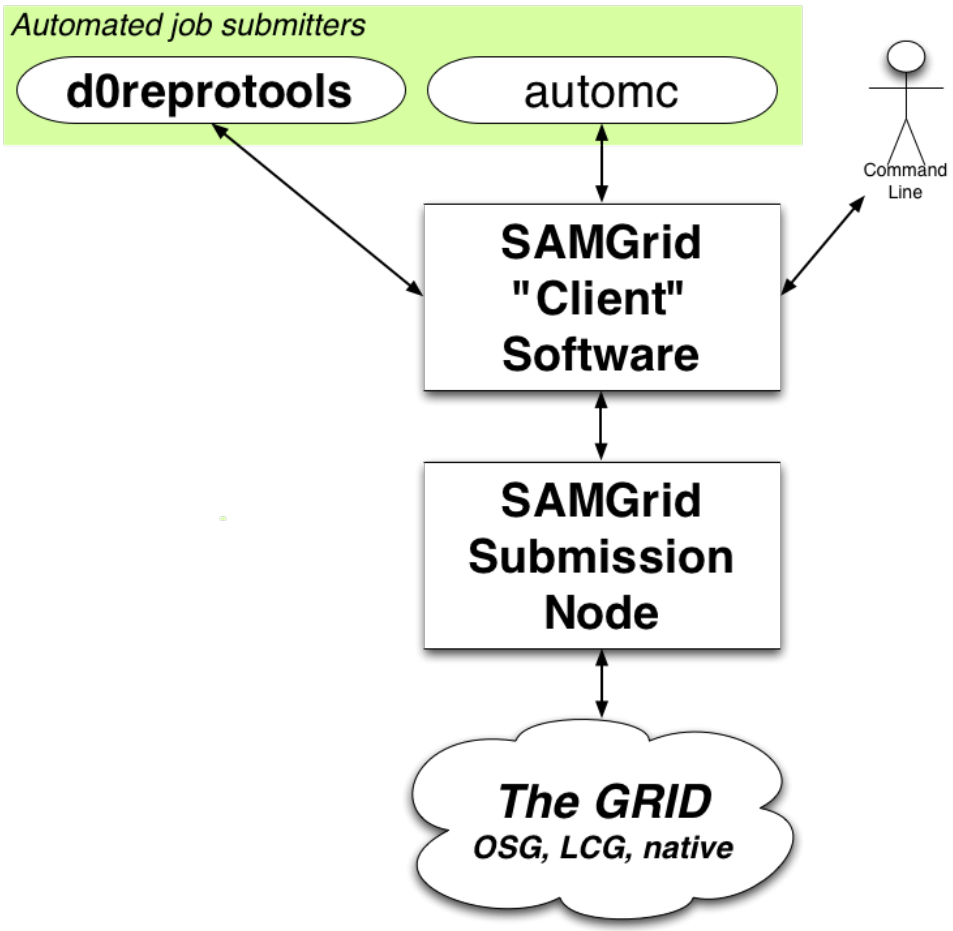

Figure 1: Components handing submission of SAMGrid jobs

The head/forwarding node also starts a SAM Data Handling project, if necessary. The SAM Project Master process resides on this head/forwarding node.

3.3 Running the Application

Once an execution job lands on a worker node, an infrastructure for running the application takes over as shown in Figure 3. The SAMGrid execution script is the overall job script. `D0RunJob` handles the workflow and knows details of the DØ applications. The SAMGrid execution script must also know details of the application in order to set up `D0RunJob`. The applications are executed by `D0RunJob`.

As seen in the figure, a job may need information from services outside of the execution site.

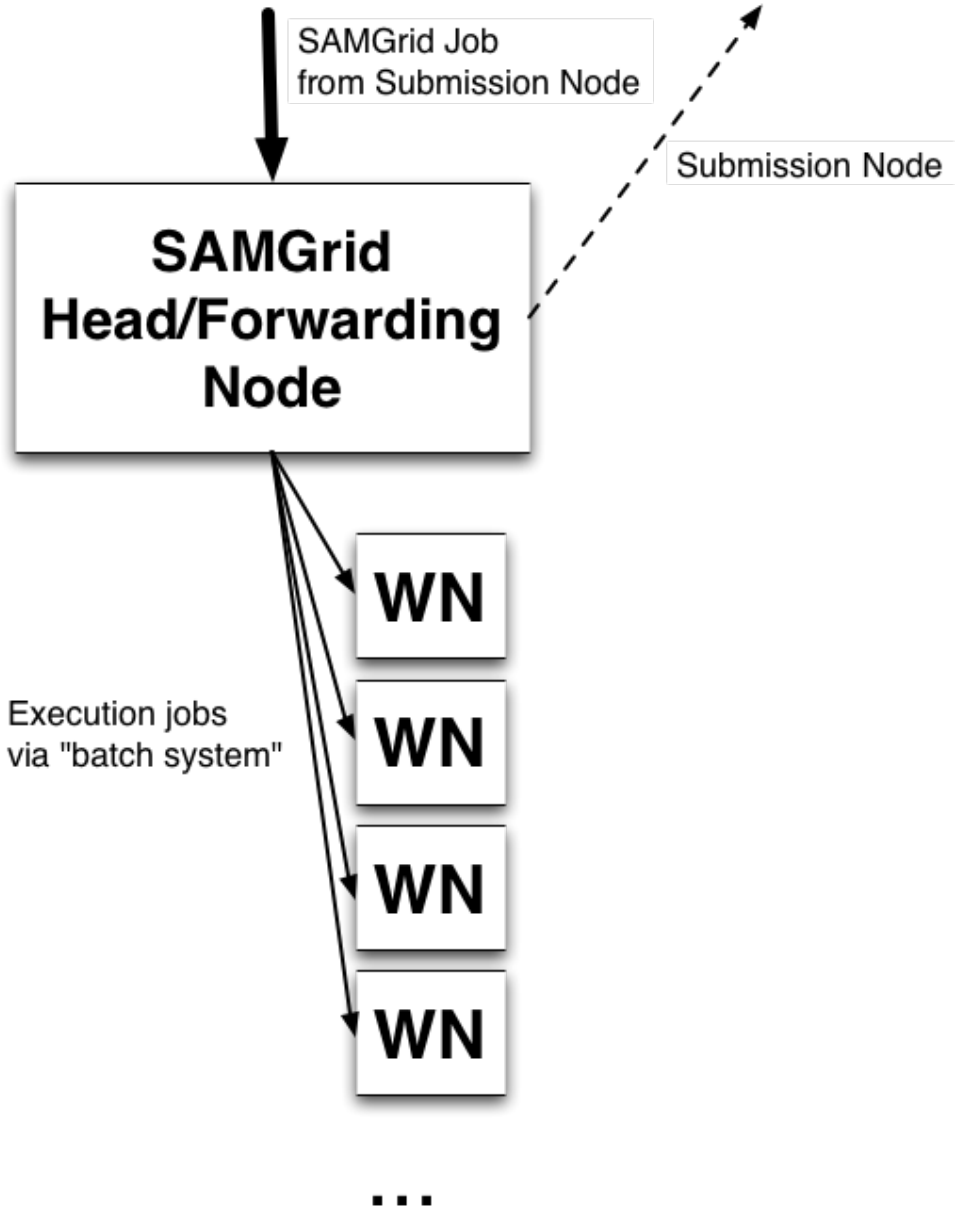

Figure 2: Components for executing jobs on the Grid

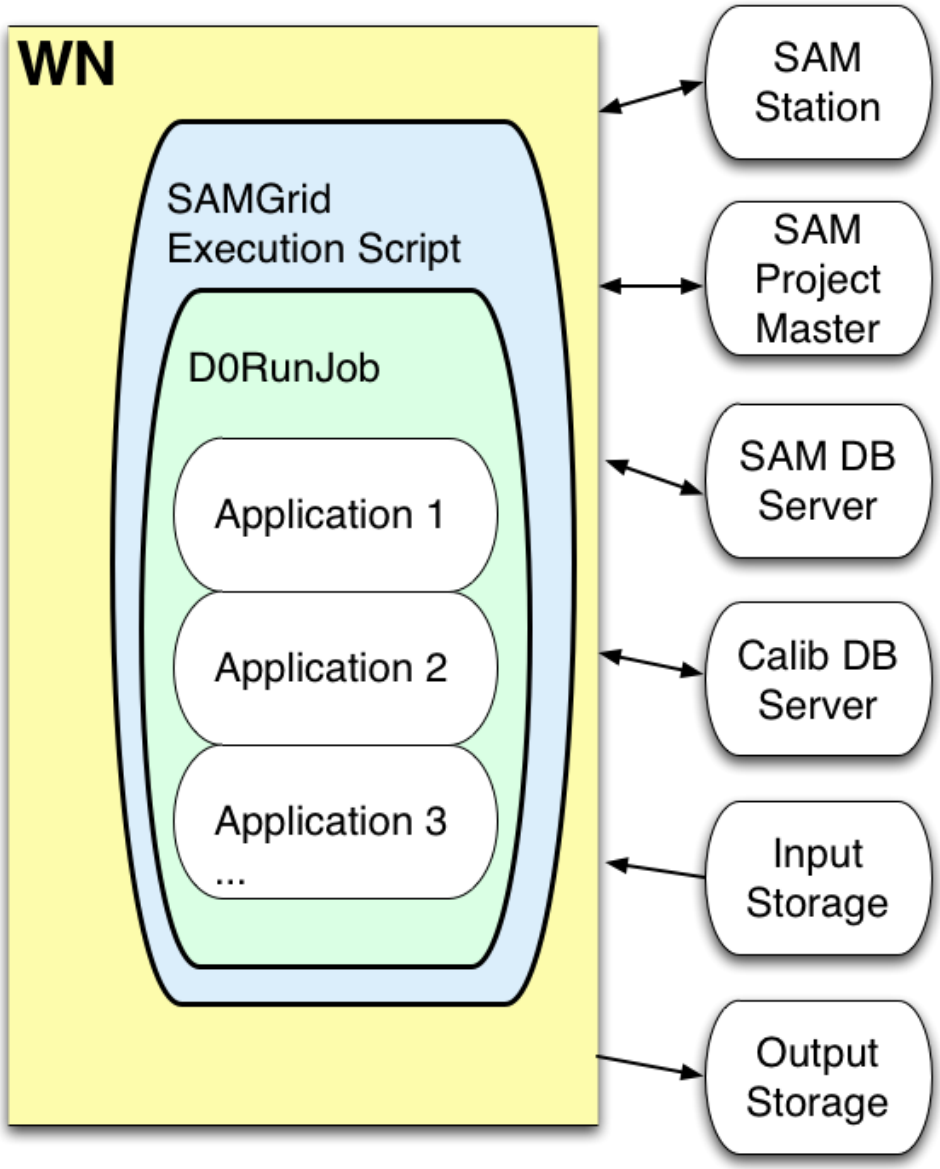

Figure 3: Components for running the Application

4 Competing Expectations

I belive there are three competing expecations when writing software and tools for production on the Grid.

1. Efficiency of operations
2. Expediency in bringing features into production
3. Ease of maintenance and use of generic tools

4.1 Efficiency of Operations

There is rarely time to spare when running production, therefore there is an expectation and indeed a demand that jobs use resources as efficiently as possible and run as quickly as possible. The SAMGrid team has learned that to make the system efficient, knowledge of the specific application needs to be embedded in various parts of the system. For example, the submission node translates a SAMGrid job into execution jobs based on the application type. The SAMGrid data handling code knows about the application in order to optimize data delivery (e.g. jobs that need to be fed data rapidly do not pull from the same data queue as jobs that need data more rarely). Adding application specific code to improve efficiency makes the system more complicated and more difficult to maintain and adapt to future needs.

4.2 Expediency in bringing features into production

Much of SAMGrid was written when standard Grid tools were in their infancy. Since DØ needed to perform production tasks quickly, we developed our own Grid system. But with the LCG and OSG forwarding nodes, we are migrating native SAMGrid execution sites to the OSG/LCG Grids.

The need to bring features into production quickly has also lead to code that is difficult to maintain, including the current division of responsibility between SAMGrid and `D0RunJob`.

Again, since there is rarely spare time when running production, the lack of certain features means that production is slower and less efficient than it would be otherwise. The DØ task list shows the extent of desired features.

4.3 Ease of maintenance and generic tools

As mentioned above, the need for high efficiency has lead to application specific code embedded in the system. The need to bring features into production quickly has lead to code that is difficult to maintain. Furthermore, most of the development effort has gone into making features work and then moving on instead of having them work easily or in a way that is easily adaptable to new ideas.

4.4 The interplay of these expectations

The ease of maintenance expectation has more frequently been traded away in favor of the other two, and now we are feeling the effects of those decisions. Can DØ forgo some efficiency in favor of a more generic system? The desire to add new features as quickly as possible is always present, but perhaps now is a good time to make other changes that improve maintainability and ease of use even of these improvements do not immediately lead to new features.

5 What can we do and how can you help?

The fact that the DØ task list is so long means (at least to me) that SAMGrid has really just begun to automate production tasks and there's much more to do. The fact that we cannot add these features easily shows a deficiency in the system.

But it is important to keep in mind that the SAMGrid system is quite mature and in constant use for production. The time scale for production at DØ is probably only another 2-3 years. Drastic changes are perhaps unwise at this point.

But can we identify some less drastic changes that could lead to meaningful improvements? Are there other experiences (CMS, CDF, ...) that can teach us what to do next? Is there code and expertise that we can utilize?